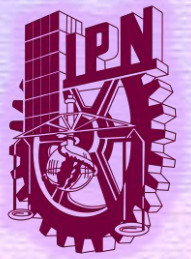


INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Aplicación para comunicaciones en red

3CM17

Practica 03

"Aplicaciones Peer to peer (P2P)"

Integrantes:

- **Bocanegra Heziquio Yestlanezi**
- **Martinez Cruz José Antonio**

Profesor: Moreno Cervantes Axel Ernesto

índice

Índice de figuras 3
Introducción 4
Servidor RMI 4
Servidor multicast 5
Dirección IP multicast 5
Distribución eficiente 6
Servidor de flujo 6
MD5Checksum 6
Usos del cifrado MD5 6
Desarrollo 7
Conclusiones 12
Bocanegra Heziquio Yestlanezi 12
Martínez Cruz José Antonio 12

índice de figuras

Figura 1 Componentes 4
Figura 2 modelo de cuatro capas 5
Figura 3 Servidor Multicast 7
Figura 4 Servidor RMI 8
Figura 5 Servidor unicast 8
Figura 6 Interfaz 9
Figura 7 No existe el archivo 9
Figura 8 Archivo encontrado 10
Figura 9 Descarga en servidor 11
Figura 10 Búsquedas 11
Figura 11 Archivos descargados 11

Introduccion

Servidor RMI

RMI (*Remote Method Invocation*) es un mecanismo que permite realizar llamadas a métodos de objetos remotos situados en distintas (o la misma) máquinas virtuales Java, compartiendo así recursos y carga de procesamiento a través de varios sistemas.
RMI permite exportar objetos como objetos remotos para que otro proceso remoto pueda acceder directamente como un objeto Java. Todos los objetos de una aplicación distribuida basada en RMI deben ser implementados en Java. Esta es una de las principales ventajas de RMI, ya que RMI forma parte del API de Java, con lo que la integración de objetos remotos en aplicaciones distribuidas se realiza sin necesidad de usar recursos adicionales (como por ejemplo un lenguaje de descripción de interfaces o IDL). De hecho, se utiliza la misma sintaxis para una llamada a un objeto remoto o un objeto local.
La siguiente figura 1 se ilustra los componentes implicados en un sistema RMI: un interfaz remoto, un cliente, y uno o más servidores (objetos remotos) residentes en un host.

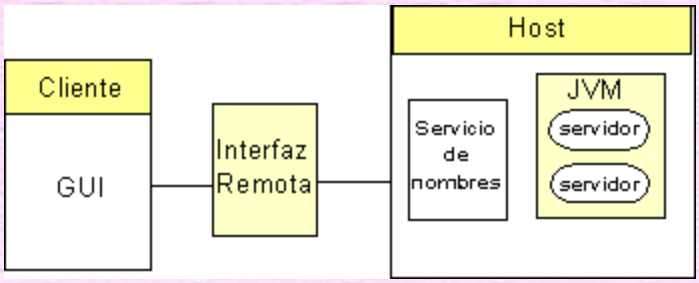


Figura 1 Componentes

El cliente invoca a los objetos remotos mediante la interfaz remota. Un **servicio de nombres** (registro RMI) reside en el *host* proporcionando el mecanismo que el cliente usa para encontrar uno más servidores iniciales RMI.
La interacción con el objeto remoto se lleva a cabo a través de la **interfaz remota**. Esencialmente, ésta describe los métodos que pueden ser invocados de forma remota, y que el objeto remoto implementa. Cuando se obtiene una referencia a un objeto remoto, el objeto no se envía a través de la red al cliente que lo solicita. En su lugar se genera un objeto *proxy* o *stub* que constituye el *proxy* de la parte del cliente del objeto remoto. Todas las interacciones del cliente se realizarán con esta clase *stub*, la cual es responsable de gestionar los datos entre el sistema local y el remoto. Muchos clientes pueden tener referencias a un único objeto remoto. Cada cliente tiene su propio objeto *stub* que representa al objeto remoto, pero dicho objeto remoto NO se replica.
En la parte del servidor, una clase *skeleton* es la responsable de gestionar las llamadas al método y los datos enviados al objeto real referenciado. Éste es el *proxy* de la parte del servidor para el objeto remoto. El sistema completo puede verse como un modelo de cuatro capas, tal y como se ilustra en la siguiente figura 2.

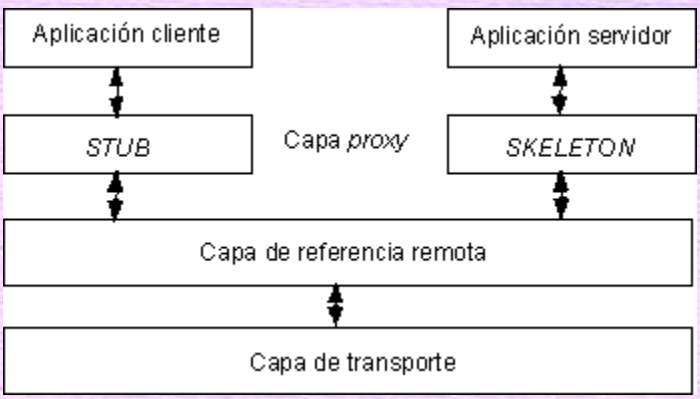


Figura 2 modelo de cuatro capas

La primera capa es la de **aplicación**, y se corresponde con la implementación real de las aplicaciones cliente y servidor. Aquí tienen lugar las llamadas a alto nivel para acceder y exportar objetos remotos. Cualquier aplicación que quiera que sus métodos estén disponibles para su acceso por clientes remotos debe declarar dichos métodos en una interfaz que extienda *java.rmi.Remote*. Dicha interfaz se usa básicamente para "marcar" un objeto como remotamente accesible. Una vez que los métodos han sido implementados, el objeto debe ser exportado. Esto puede hacerse de forma implícita si el objeto extiende la clase UnicastRemoteObject (paquete *java.rmi.server*), o puede hacerse de forma explícita con una llamada al método exportObject() del mismo paquete.

La capa 2 es la capa ***proxy***, o capa stub-skeleton. Esta capa es la que interactúa directamente con la capa de aplicación. Todas las llamadas a objetos remotos y acciones sobre sus parámetros y retorno de objetos tienen lugar en esta capa.

La capa 3 es la de **referencia remota**, es responsable del manejo de la parte semántica de las invocaciones remotas. También es responsable de la gestión de la replicación de objetos y realización de tareas específicas de la implementación con los objetos remotos, como el establecimiento de las persistencias semánticas y estrategias adecuadas para la recuperación de conexiones perdidas. En esta capa se espera una conexión de tipo *stream* (*stream-oriented connection*) desde la capa de transporte.

La capa 4 es la de **transporte**. Es la responsable de realizar las conexiones necesarias y manejo del transporte de los datos de una máquina a otra. El protocolo de transporte subyacente para RMI es JRMP (*Java Remote Method Protocol*), que solamente es "comprendido" por programas Java.

Servidor multicast

es un componente de red que permite la distribución eficiente de datos a múltiples receptores en una red. A diferencia del envío unicast (uno a uno) y del envío broadcast (uno a todos), el multicast permite la transmisión de datos a un grupo selecto de receptores interesados en recibir la información.

Direccion IP multicast: El servidor multicast utiliza una dirección IP multicast para enviar los datos al grupo de receptores interesados. Las direcciones IP multicast están en 228.1.1.1. Estas direcciones están reservadas específicamente para el uso de multicast.

Distribucion eficiente: El servidor multicast envía los datos de manera eficiente a través de la red, ya que solo se transmiten una vez desde la fuente y luego se replican en los enrutadores de la red para llegar a los receptores interesados. Esto reduce significativamente el ancho de banda necesario y la carga de la red en comparación con el envío unicast tradicional.

Servidor de flujo

es un componente de red que se encarga de distribuir datos multimedia, como audio y video, a través de una red de manera continua y en tiempo real. A diferencia de las descargas tradicionales, en las cuales se necesita esperar a que se descargue un archivo completo antes de reproducirlo, el streaming permite la reproducción inmediata de contenido multimedia a medida que se va recibiendo.

MD5Checksum

MD5 (Message Digest Algorithm 5) es un algoritmo que se utiliza como una función de codificación o huella digital de un archivo. De esta forma, a la hora de descargar un determinado archivo como puede ser un instalador, el código generado por el algoritmo, también llamado hash, viene “unido” al archivo. Un hash MD5 está compuesto por 32 caracteres hexadecimales y una codificación de 128 bits.

El código MD5, existe un software que analiza el archivo descargado y con el hash de la descarga, acudimos a la web del desarrollador del programa que queremos instalar y buscamos el código original, comprobando si coincide el que nos ha dado el hash con el que aparece en la web. De esta forma veremos si el archivo es fiable o no.

Usos del cifrado MD5

Además de para asegurarnos si nos estamos descargando e instalando un sofware fiable y no malicioso, el cifrado MD5 tiene otros usos que tal vez te interesen:

- A través de un programa podemos crear un código MD5 para alguno de nuestros archivos, y de esta manera asegurarnos que únicamente nosotros podemos hacer uso de él.
- Una vez hayamos realizado una descarga y dispongamos del archivo completo, podemos utilizarlo en una instalación de firmware como puede ser un router.
- Comprobar que un texto no ha sido modificado a la hora de enviárselo a otra persona y pueda llegar de forma distinta a como era el original. Existen páginas webs en donde nosotros metemos el texto que vamos a enviar, esta web nos devuelve el hash, y es este el que enviamos a nuestro destinatario para que compruebe que el texto es el correcto.

Desarrollo

Como se mencionó al inicio para esta práctica se hace uso de 3 diferentes tipos de servidores corriendo en 3 hilos distintos, uno es el servidor Multicast en donde residen los otros 2 servidores, además de ser en donde los clientes se conectan para poder hacer la búsqueda y la descarga de los archivos en el servidor. El servidor RMI nos sirve para poder realizar la búsqueda de los archivos en nuestro servidor, además de que ahí también hacemos uso del MD5Checksum y finalmente el servidor unicast para poder hacer él envió de nuestro archivo encontrado si es lo que solicito el cliente.

Podemos observar en la figura 3 el servidor Multicast ejecutándose

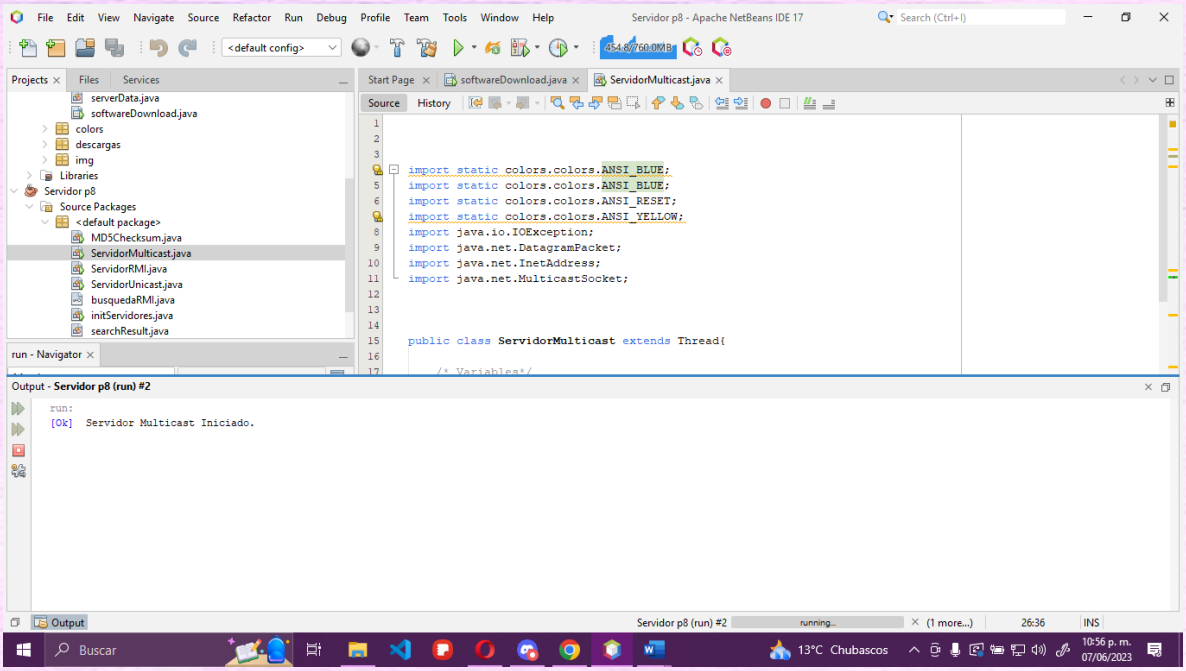


Figura 3 Servidor Multicast

Ahora podemos observar en la figura 4 el servidor RMI ejecutándose

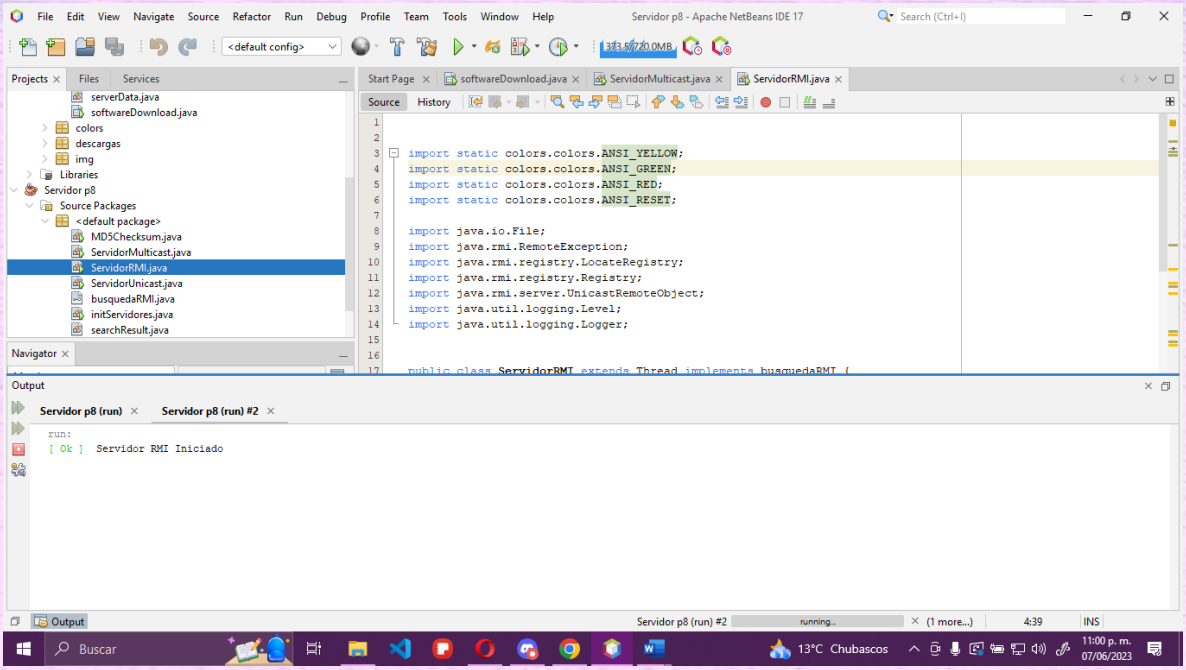


Figura 4 Servidor RMI

Y como tercer servidor podemos observar en la figura 5 iniciado el servidor uniticast

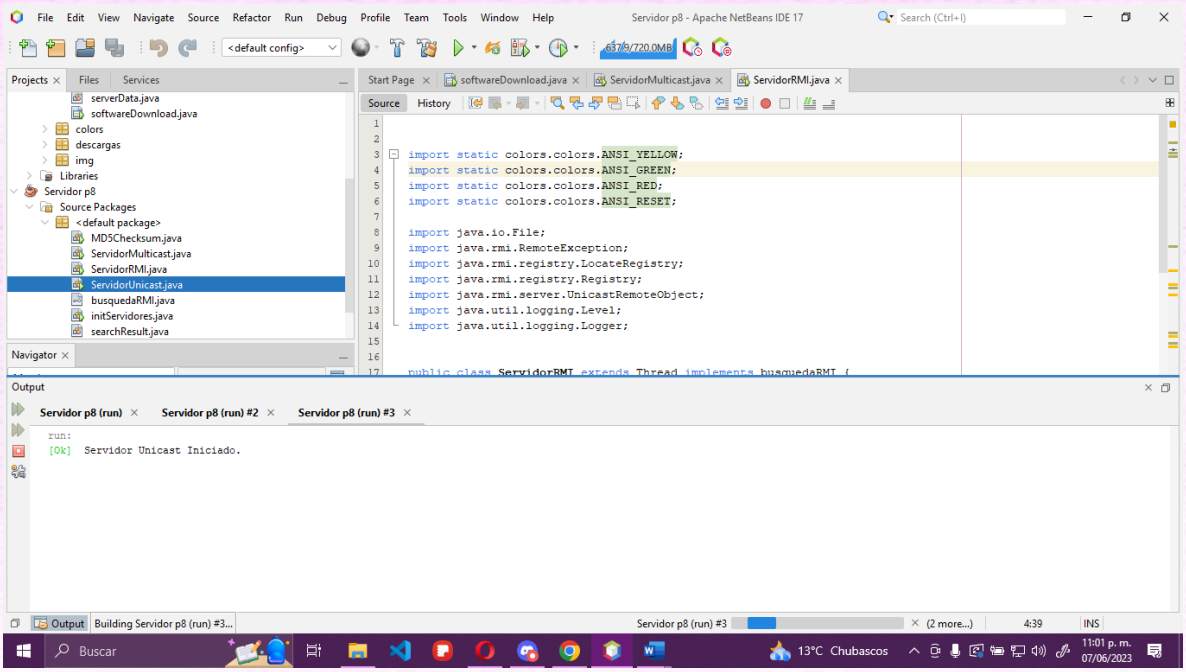


Figura 5 Servidor unicast

Por último, en la figura 6 podemos observar la interfaz para la búsqueda de archivos, la cual inicializa los tres servidores.

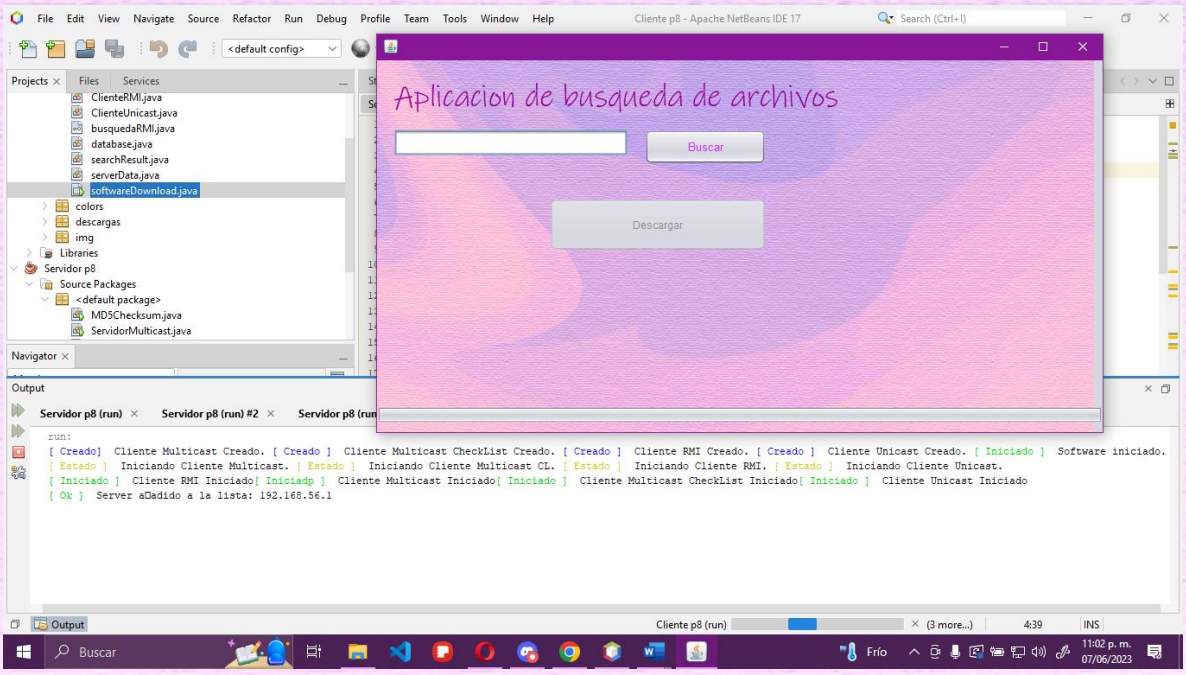


Figura 6 Interfaz

Como podemos observar en la figura 7 al momento de realizar una búsqueda de la carpeta de database, debemos escribir el nombre tal cual esta en la carpeta, incluso con el .png del formato de la imagen, ya que de otra forma no lo encontrara como se muestra.

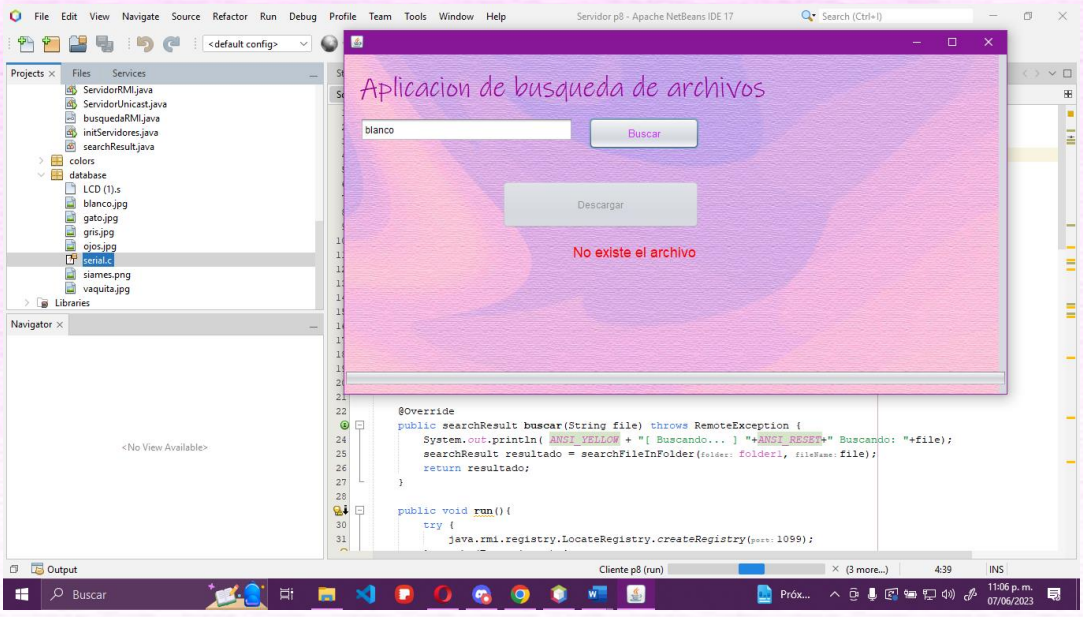


Figura 7 No existe el archivo

En la figura 8 podemos observar que encuentra el archivo si se escribe exactamente igual a como aparece en la database

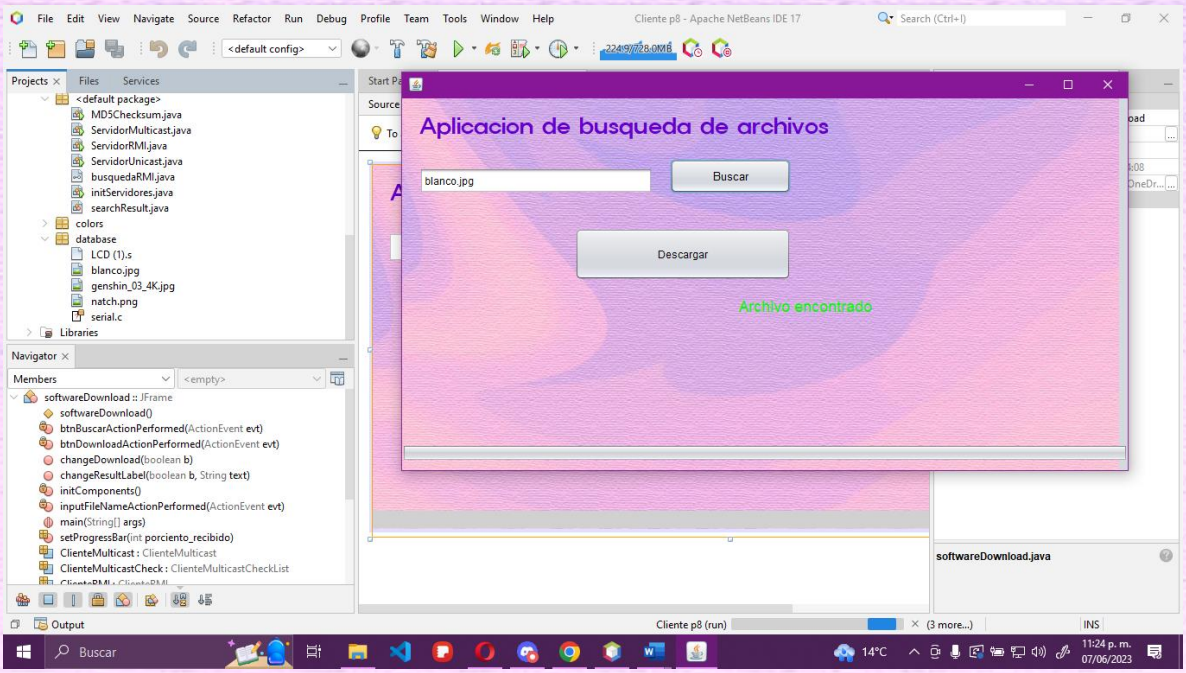
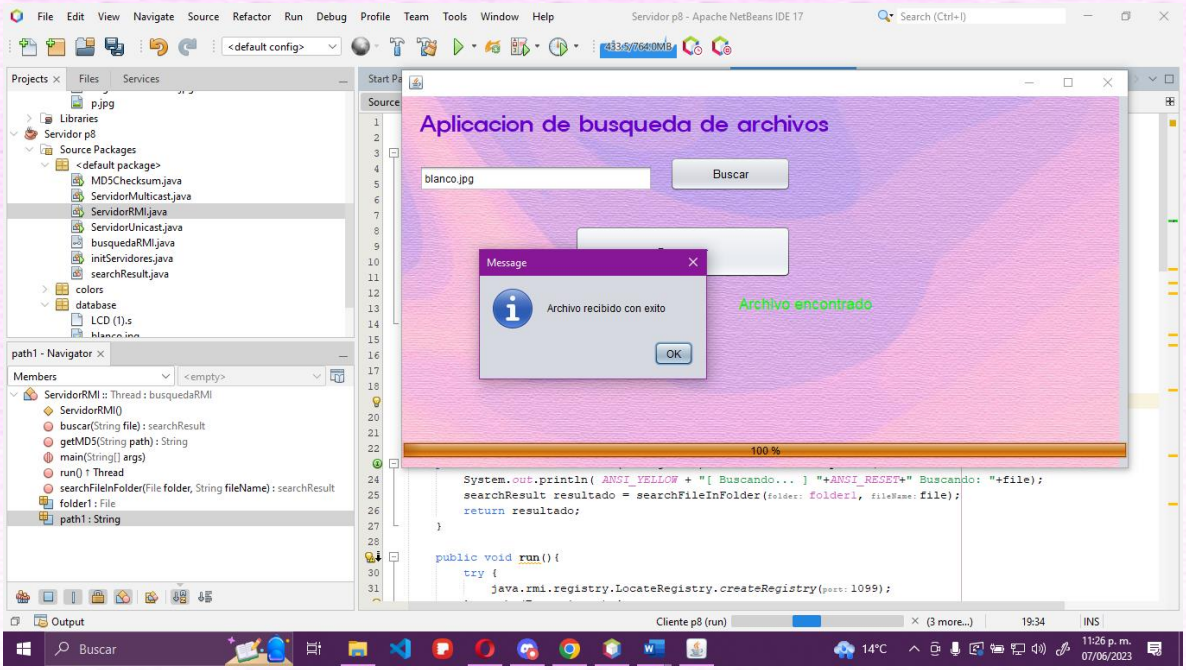


Figura 8 Archivo encontrado

Después en la figura 9 podemos observar el mensaje mostrado cuando tratamos de descargar el archivo



Y en el servidor podemos observar lo siguiente

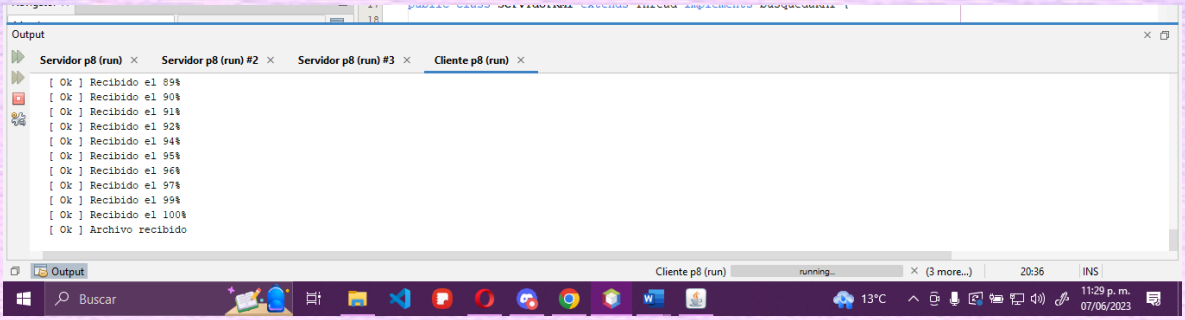


Figura 9 Descarga en servidor

Así como en el servidor RMI podemos observar las búsquedas realizadas

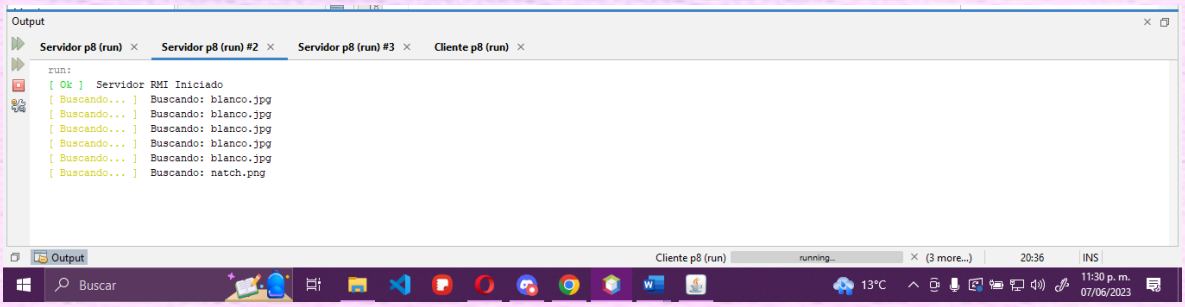


Figura 10 Búsquedas

Finalmente podemos observar en la figura 11 en la carpeta establecida la descarga de los archivos seleccionados

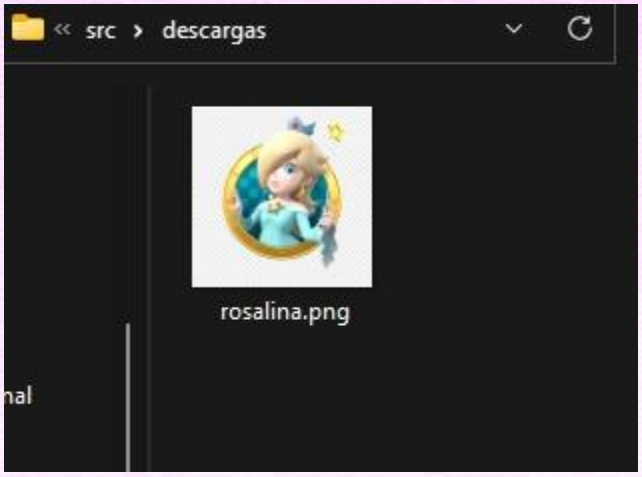


Figura 11 Archivos descargados

Conclusiones

Bocanegra Heziquio Yestlanezi

En esta práctica se puede observar de manera perfecta el uso de servidores RMI, multicast y unicast, prácticamente usando todo lo visto en el curso, lo cual es muy interesante, ademas, de que la aplicación que se desarrollo es una aplicación muy famosa y usada, permite el rápido entendimiento de estos temas y ademas de ser entretenido el desarrollo, agregar que el hecho de hacer uso de MD5Checksum es interesante ya que vemos un añadido bastante interesante e importante, el cual muy posiblemente será útil para futuros programas.

Martínez Cruz José Antonio

El desarrollo de esta práctica resultó ser el más complejo que se ha tenido en esta asignatura y nos demoró bastante tiempo lograr los objetivos establecidos. Sin embargo, sus funciones se realizan de manera apropiada y congruente, tal como la consulta, búsqueda y descarga del archivo solicitado mediante el uso de tres servidores que desempeñaban los servicios multicast o unicast.